IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of: Bates et al.	§	Confirmation No.: 3890
	§	
	§	Group Art Unit: 2193
Serial No.: 10/090,341	§	
	§	Examiner: Mark P. Francis
Filed: March 4, 2002	§	
	§	
For: DEBUG OF CODE WITH SELECTIVE DISPLAY OF DATA	§	

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450, or facsimile transmitted to the U.S. Patent and Trademark Office to fax number 571-273-8300 to the attention of Examiner Mark P. Francis, or electronically transmitted via EFS-Web, on the date shown below:

June 20, 2007 /Jon K. Stewart/
Date Jon K. Stewart

APPEAL BRIEF

Dear Sir:

Applicants submit this Appeal Brief to the Board of Patent Appeals and Interferences on appeal from the decision of the Examiner of Group Art Unit 2193 dated October 20, 2006, finally rejecting claims 1, 2, 4-6, 8-12, and 14-20. Applicants appeal the final rejection of claims 1, 2, 4-6, 8-12, and 14-20. This Appeal Brief is believed to be timely since it is electronically transmitted by the due date of June 20, 2007, as set by mailing a Notice of Appeal on April 20, 2007. Please charge the fee of $500.00 for filing this brief to Deposit Account No. 09-0465/ROC920010348US1.

TABLE OF CONTENTS

Real Party in Interest

The present application has been assigned to International Business Machines Corporation, Armonk, New York.

Related Appeals and Interferences

Applicant asserts that no other appeals or interferences are known to the Applicant, the Applicant's legal representative, or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

Status of Claims

Claims 1-2, 4-6, 8-12 and 14-20 are pending in the application. Claims 1-20 were originally presented in the application. Claims 3, 7 and 13 have been canceled without prejudice. Claims 1, 2, 4-6, 8-12 and 14-20 stand finally rejected as discussed below. The final rejection of claims 1-2, 4-6, 8-12 and 14-20 is appealed. The pending claims are shown in the attached Claims Appendix.

Status of Amendments

Amendments to the claims were proposed after the final rejection. All claim amendments have been entered by the Examiner.

Summary of Claimed Subject Matter

Claimed embodiments include provide a method (see e.g., claim 1), and computer readable medium (see e.g., 9 and 16) for debugging executable code configured to access associated data in a data repository. See e.g., *Application*, 2: 2-4, 3: 22-30, 5: 13-16, Abstract. That is, the executable code being debugged includes routines that access data stored in a data repository. *Id.*

Claimed embodiments include a method (see e.g., claims 1, 2, 4-6, 8) and a computer readable medium (see e.g., claims 9-11, 13-4) for selectively displaying data during debugging. See *Application*, 3: 22-30, 5: 13-16, 11: 30-31, 12: 1-31, 13: 1-18, Figure 8. The claimed embodiments include initiating a debugging session for the executable code. See e.g., *Application*, 3: 22-30, 8: 12-21, 12: 17-27, Figure 8. Once initiated, the method includes monitoring the step-by-step execution of the executable code. See e.g., *Application*, 3: 22-30, 8: 22-29, 9: 1-20, 11: 9-29, 13: 3-8, Figure 8, 804-818. The method also includes determining whether the monitored executable code has accessed the associated data in the data repository. See e.g., *Application* 3: 22-30, 13: 9-18, Figure 8, 810. If so, the method includes, determining whether to display the associated data on the basis of whether the associated data is restricted data. See e.g., A*pplication*, 3: 22-30, 9: 21-29, 10: 1-14, 13: 9-18, Figure 4, 144, Figure 8, 818. As claimed, the method also includes determining whether to display the associated data comprises referencing predefined access restriction rules defining at least one rule preventing at least a portion of the associated data from being displayed to unauthorized users. *Id.* If it is determined to not display the associated data on the basis of the referenced predefined access restriction rules, the method includes outputting masking characters on an output screen indicative of the associated data without revealing a value of the associated data, whereby selected data from the data repository is concealed from a user debugging the executable code. See e.g., *Application*, 7: 16-31, 11: 30-31, 12: 1-16, 9: 21-29, 10: 15-28, 10: 29-31, 11: 1-8, 13: 9-18, Figure 8, 820.

Claimed embodiments also include a computer readable medium (see e.g., claims 17-20) containing a debug program which, when executed, performs an operation of debugging code. The code is configured to access associated data in a repository. The claimed debug program is configured to provide debugger user interface. See *Application*, 3: 22-30, 5: 13-16, 8: 12-29, 9: 1-20, 11: 30-31, 12: 1-31, 13: 1-18, Figure 4, Figure 8. The debug program is also configured with a debug engine configured to selectively pass data to the debugger user interface according to predefined access restriction rules defining at least one rule prohibiting at least a portion of the associated data from being displayed to a user operating the debug program, whereby selected data from the data repository is concealed from the user debugging the executable code. See e.g., *Application*, 3: 22-30, 7: 16-31, 9: 21-29, 10: 1-31, 11: 1-8, 11: 30-31, 12: 1-31, 13: 1-18, Figure 8.

Grounds of Rejection to be Reviewed on Appeal

1. Claims 1-2, 4-6, 8-12 and 14-20 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Wimble,* U.S. Pat. No. 5,812,850 (hereinafter *Wimble*) in view of *Warmink et al.*, U.S. Pat. No. 6,611,924 (hereinafter *Warmink*).

ARGUMENTS

Obviousness of claims 1-2, 4-6, 8-12 and 14-20 over *Wimble* in view of *Warmink*.

The Examiner bears the initial burden of establishing a *prima facie* case of obviousness. *See* MPEP § 2142. To establish a *prima facie* case of obviousness three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one ordinary skill in the art to modify the reference or to combine the reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *See* MPEP § 2143. The present rejection fails to establish at least the third criteria.

More specifically, Applicants submit that *Wimble* does not disclose "a method of debugging executable code configured to access associated data in a data repository" that includes "determining whether the monitored executable code has accessed the associated data in the data repository," as recited by claim 1. Claim 9 recites similar limitations. Further, *Warmink* does not disclose "a method of debugging executable code configured to access associated data in a data repository" that includes "upon determining not to display the associated data on the basis of the referenced predefined access restriction rules, outputting masking characters on an output screen indicative of the associated data without revealing a value of the associated data, whereby selected data from the data repository is concealed from a user debugging the executable code," as recited by claim 1. Claim 9 recites similar limitations.

The References

Wimble discloses "a debugging system which provides an interactive and dynamic environment for computer program debugging." *Wimble*, 1:18-20. "The debugging system uses a database of information relating machine executable code to source code. The database is developed during the compilation process using an

extensible object-oriented set of tools." *Wimble*, Abstract. The debugger disclosed in *Wimble*

> generates *information to be used by the debugger* in a Debugging System. The information may be in the form of data bases. Debugging information is really a database of information about a compiled program.

Wimble, 6:30-34. *Wimble* is very specific regarding the information stored in the debugger database. Specifically, *Wimble* discloses that information in the debugger database includes "a collection of Symbolic Elements 417 which is to be built, and maintain *a description of the program once it has been built*. *Wimble*, 8:27-37. As the emphasized passage makes clear, the debugger database of *Wimble* is created and used by the debugger to assist in the debugging process, and is not accessed by the program being debugged.

Warmink discloses a process where the developer is presented with the original debug message. *Warmink* describes the process of replacing debug strings in the source code of a program with unique numbers (referred to as ENUM values) as follows:

> Although the text message is the largest portion of the trace statement's argument, it is also fixed in length and content for each trace statement. The present invention exploits this fact by removing or stripping out all of these text message or fixed field portions of debug trace statement arguments, and replacing them with unique numbers. Thus, source code is first obtained (step 201). This is typically the final version of the source code after development phase debugging, just before the code is ready for final compiling and shipping to end users.
>
> Next, the source code is run through a debug strip tool or program in accordance with the present invention, to identify each trace statement and its fixed text string portion. This debug strip program may be run on a PC (e.g., the same PC on which programming and debugging has been performed to write the program code) and provided with the file name of the completed source code. Each of the fixed text strings in the argument portions of trace statements in the source code is replaced with a unique number, such as an ENUM (step 203). This involves deleting the text string, inserting the next ENUM in the sequence (and incrementing the ENUM to the next one as necessary), and saving both the text string and the ENUM corresponding to that text string. The variable data portions of

> the output string are not stripped out but remain, with the ENUM, in the output string.

Warmink, 6:35-60. This passage describes a process of identifying text-string debugging messages in the source code of a program and replacing the text strings with "a unique number." The unique number does not prevent or conceal the debugging message, in fact the opposite is the case. *Warmink* goes on to describe how the "unique numbers" are used to generate debug messages presented to a developer. When a "debug PC" is attached to a "debug port." Specifically, *Warmink* teaches:

> the "debug pc converts the debug value to a corresponding text string using a "number to string mapping table. The text string, not the ENUM value, is thus inserted in the appropriate place in the debug output string of data and displayed on monitor 126 along with other data of the debug output string.

7:65-67 – 8:1-3. In other words, the original debug string message is displayed to the user.

Despite what *Wimble* and *Warmink* disclose, the Examiner continues to assert that *Wimble* discloses steps of a method of debugging executable code, where the executable code being debugged is configured to access associated data in a data repository, in the manner claimed. To support the present rejection, the Examiner suggests that *Wimble* discloses "determining whether the monitored executable code has accessed the associated data in the data repository (Col12:47-67, '... a component in the Debugger Database ...')" *Final Office Action*, p. 3. Respectfully, the Examiner's argument is untenable, as there is no readily apparent relationship between "a component in the debugger database" and the claimed method step of "determining whether the monitored executable code has accessed the associated data in the data repository." Further, the program being debugged using the System of *Wimble* would not even access the debugging information used by the debugger. Set out in full, the cited passage provides:

> ADOPTPCSOURCEMAP
>
> FIG. 11 is a block diagram of the function AdoptPCSourceMap 129 for asking the information reader to adopt a particular format of information from a provider. The provider must put together a list of

PCSourceMapElement entries 130 and ask the Debugger to adopt the information which describes a component in the Debugger Database. The following shows the Adopt function interface:

---

void AdoptPCSourceMap (const THoopsPropertyName name, TPCSourceMapList* pcSourceMap);

There is a different map kept to describe the Interface v. the Implementation properties of a component, so the developer must supply a "name" parameter to indicate which property the map describes.

THE TOKEN MAP

FIG. 12 is a block diagram showing a general overview of Token Map 66, and details of the Token Map entry objects 78.

Plainly, nothing in this passage discloses the step of determining whether the code being debugged accessed data from a data repository. Instead, the passage describes the use of function call "AdoptPCSourceMap" to ask an "information reader" to adopt a particular format of information from a provider. For all these reasons, Applicants submit that while *Wimble* discloses a debugger configured to create a database of debugging information, this fails to disclose the recited limitation of debugging executable code that is itself *configured to access associated data in a data repository*.

In addition to the passages from *Wimble* cited in the Final Office Action, the Examiner suggests in the *Advisory Action,* continuation sheet, that "Wimble does disclose the executable code accessing the associated data from the data repository" at Col. 8, lines 16-49. However, Applicants submit that *Wimble* does not disclose this. *Wimble,* Col. 8, lines 16-49, only refer to components created "under direction of the Debugger 48, in the same Database 41." Therefore, the changes that they undergo are completely unrelated to "determining whether the code being debugged accessed data from a data repository" (emphasis added).

Further, it directly follows that *Wimble* does not disclose the recited limitation of determining whether the monitored executable code has accessed the associated data in the data repository, as it would make no sense for the system of *Wimble* to do so. The debugger of *Wimble* would not create the debugger database and then monitor

whether the very same debugger accesses the debugger database. Access by the debugger to the debugger database would be presumed. Otherwise, the debugger would not bother to create it. On this basis alone, Applicants submit that the rejection should be withdrawn and the claims be allowed.

Furthermore, the Examiner concedes that *Wimble* fails to disclose the claimed steps of:

> if so, determining whether to display the associated data on the basis of whether the associated data is restricted data; wherein determining whether to display the associated data comprises referencing predefined access restriction rules defining at least one rule preventing at least a portion of the associated data from being displayed to unauthorized users; and
>
> upon determining not to display the associated data on the basis of the referenced predefined access restriction rules, outputting masking characters on an output screen indicative of the associated data without revealing a value of the associated data, whereby selected data from the data repository is concealed from a user debugging the executable code.

Claim 1. Claim 9 recites a similar limitation. In other words, if a program being debugged accesses data from a data repository that is restricted data (i.e., person debugging the program is not authorized to view certain data from the database), this information may be masked using masking characters. The material cited by the Examiner refers to debugging messages that may be "selectively enabled based on developer-selected masks, based on some specified rules of filtering." *Warmink*, 2:39-40.

However, as used in *Warmink*, a numerical value—referred to as a "mask value"— is used as a reference to messages output during program execution. No use of masking characters is disclosed, or would be useful using this debugging technique. That is, it makes no sense whatsoever to suggest that the debugging messages, meant to inform the programmer of aspects of program operational state, could be replaced with masking characters when displayed to a user and remain useful at all. Thus, Applicants contend that the "mask value" used in *Warmink* to filter which debug messages are displayed on a "debug PC" in no way discloses the claimed method step

of "outputting masking characters on an output screen indicative of the associated data without revealing a value of the associated data, whereby selected data from the data repository is concealed from a user debugging the executable code." Plainly, nothing is concealed, and no part of the debug message is the developer restricted from viewing.

Nevertheless, in the *Advisory Action,* continuation sheet, the Examiner suggests that "*Warmink* does teach outputting masking characters on an output screen indicative of the associated data without revealing a value of the associated data, whereby selected data from the data repository is concealed from a user debugging the executable code" at Col 2:40-65. However, the cited passage teaches that "the use of such debug trace statements allows the program developer to specify and thus enable only the subset of debug messages which are of interest." Whichever messages are output, are done so without being "masked" or otherwise concealed from the developer at all. Therefore, not only is it the developer who specifies what "messages" are of interest, but the "messages" are "debug messages". It is clear from the passage that there is no mention of any sort of filtering or "masking" of the "associated data."

Accordingly, for all the foregoing reasons, Applicants submit that claims 1, 9, and the claims dependent therefrom, are patentable over *Wimble* in view of *Warmink*.

Regarding claim 16:

The Examiner suggests that *Wimble,* in view of *Warmink*, discloses a computer-readable medium containing a debug program which, when executed, performs an operation of debugging code configured to access associated data in a repository. However, Applicants contend that *Wimble*, in view of *Warmink*, fails to disclose at least the recited limitation of "a debug engine configured to selectively pass data to the debugger user interface according to predefined access restriction rules defining at least one rule prohibiting at least a portion of the associated data from being displayed to a user operating the debug program, whereby selected data from the data repository is concealed from the user debugging the executable code."

The Examiner concedes that *Wimble* fails to disclose this limitation, but asserts that *Warmink* does. For all the reasons given above, however, Applicants submit that the passages cited from *Warmink* fail to teach or suggest a debugger interface configured to prohibit "at least a portion of the associated data from being displayed to a user operating the debug program, whereby selected data from the data repository is concealed from the user debugging the executable code." Rather, as discussed above, *Warmink* discloses a debugger user interface that allows a numerical value to filter which embedded debug messages (or codes representing such messages) are output during the execution of a program. Plainly, the debugging messages are not selected data from a data repository, as the debugging messages are embedded within the program code. Accordingly, Applicants assert that *Wimble*, in view of *Warmink*, fails to teach or suggest the limitations recited by claim 16.

Further, in the *Advisory Action*, continuation sheet, the Examiner suggests that *Warmink* "does teach a debug engine configured to selectively pass data to the debugger user interface according to predefined access restriction rules defining at least one rule prohibiting at least a portion of the associated data from being displayed to a user operating the debug program, whereby selected data from the data repository is concealed from the user debugging the executable code" at Col 2:35-67 and Col 4:40-65. However, as has already been established, lines 35-67, do not disclose "prohibiting at least a portion of the associated data from being displayed to a user operating the debug program." Lines 35-67 disclose a debugger user interface that allows a numerical value to represent embedded debug messages output during the execution of a program. Furthermore, Col 4:40-65 do not refer to the limitation at hand and only discuss the hardware used in a preferred embodiment.

Therefore, for all the foregoing reasons, claims 1, 9, 16, and the claims dependent therefrom are believed to be allowable, and allowance of these claims is respectfully requested.

Claims 8, 14 and 20 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Wimble* in view of *Warmink* and further in view of *Kolawa* (U.S. Pat. 6,085,029).

Claims 8, 14, and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Wimble* and *Warmink* in further view of *Kolawa* (U.S. 6,085,029). Applicants respectfully traverse this rejection. Claims 8, 14, and 20 depend from one of claims 1, 9 or 16, and are therefore believed to be allowable for the reasons provided above. Accordingly, withdrawal of this rejection is respectfully requested.

Therefore, for all of the foregoing reasons, the claims are believed to be allowable, and allowance of the claims is respectfully requested.

CONCLUSION

The Examiner errs in finding that claims 1-2, 4-6, 8-12 and 14-20 are unpatentable over *Wimble* in view of *Warmink* under 35 U.S.C. § 103(a). Withdrawal of the rejection and allowance of all claims is respectfully requested.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

/Gero G. McClellan, Reg. No. 44,227/

Gero G. McClellan
Registration No. 44,227
Patterson & Sheridan, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Appellants

CLAIMS APPENDIX

1. (Previously Presented) A method of debugging executable code configured to access associated data in a data repository, comprising:

initiating a debugging session for the executable code, and, during the debugging session:

monitoring step-by-step execution of the executable code;

determining whether the monitored executable code has accessed the associated data in the data repository;

if so, determining whether to display the associated data on the basis of whether the associated data is restricted data; wherein determining whether to display the associated data comprises referencing predefined access restriction rules defining at least one rule preventing at least a portion of the associated data from being displayed to unauthorized users; and

upon determining not to display the associated data on the basis of the referenced predefined access restriction rules, outputting masking characters on an output screen indicative of the associated data without revealing a value of the associated data, whereby selected data from the data repository is concealed from a user debugging the executable code.

2. (Previously Presented) The method of claim 1, wherein determining whether to display the associated data comprises determining whether the associated data can be provided to a debugger user interface.

3. (Canceled)

4. (Previously Presented) The method of claim 1, wherein determining whether to display the associated data comprises referencing a restricted data table created in response to reading the associated data from the repository and according to the predefined access restriction rules.

5. (Previously Presented) The method of claim 1, wherein determining whether to display the associated data is performed by a debugging program.

6. (Previously Presented) The method of claim 1, wherein determining whether to display the associated data is performed by a debugging program implementing the predefined access restriction rules.

7. (Canceled)

8. (Previously Presented) The method of claim 1, wherein determining whether to display the associated data comprises referencing a parse expression defining a data format and an output expression defining a restricted portion of the parse expression.

9. (Previously Presented) A computer-readable storage medium containing a debug program which, when executed, performs an operation comprising, during debugging of executable code:

monitoring step-by-step execution of the executable code, wherein the executable code is configured to access associated data in a repository;

determining whether the executable code has accessed the associated data in the repository; and

determining whether the associated data can be displayed on the basis of whether the associated data is restricted data; wherein determining whether the associated data can be displayed comprises referencing predefined access restriction rules defining at least one rule preventing at least a portion of the associated data from being displayed to unauthorized users, whereby selected data from the data repository is concealed from a user debugging the executable code.

10. (Original) The computer-readable medium of claim 9, wherein determining whether the associated data can be displayed comprises determining whether the associated data can be provided to a debugger user interface.

11. (Original) The computer-readable medium of claim 9, further comprising:

determining that the associated data cannot be displayed; and

outputting text characters on an output screen indicative of the associated data without revealing a value of the associated data.

12. (Previously Presented) The computer-readable medium of claim 9, wherein determining whether the associated data can be displayed comprises referencing a restricted data table created in response to reading the associated data from the repository and according to the predefined access restriction rules.

13. (Canceled)

14. (Original) The computer-readable medium of claim 9, wherein determining whether the associated data can be displayed comprises referencing a parse expression defining a data format and an output expression defining a restricted portion of the parse expression.

15. (Previously Presented) The computer-readable medium of claim 9, wherein the executable code accesses the associated data comprising a record and wherein determining whether the associated data can be displayed comprises:

referencing the predefined access restriction rules defining at least one rule preventing at least one field value from being displayed; and

determining whether the record contains the at least one field value.

16. (Previously Presented) A computer-readable storage medium containing a debug program which, when executed, performs an operation of debugging code configured to access associated data in a repository, the debug program comprising:

a debugger user interface; and

a debug engine configured to selectively pass data to the debugger user interface according to predefined access restriction rules defining at least one rule prohibiting at least a portion of the associated data from being displayed to a user

operating the debug program, whereby selected data from the data repository is concealed from the user debugging the executable code.

17. (Previously Presented) The computer-readable medium of claim 16, wherein the debug engine is configured to:

determine that the associated data cannot be displayed during the debugging session; and

conceal the display of the associated data by displaying text characters on an output screen indicative of the associated data without revealing a value of the associated data.

18. (Original) The computer-readable medium of claim 16, wherein the debug engine is configured to selectively pass data to the debugger user interface by referencing a restricted data table created in response to reading the associated data from the repository and according to the predefined access restriction rules.

19. (Original) The computer-readable medium of claim 16, wherein the at least one rule defines a value and an associated value, wherein if the associated value has been displayed the debug engine will not provide the value to the debugger user interface for display.

20. (Original) The computer-readable medium of claim 16, wherein the at least one rule defines a parse expression defining a data format and an output expression defining a restricted portion of the parse expression, whereby all values having restricted portion will not be provided to the debugger user interface for display.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.